Implementación de métodos computacionales TC2037

Actividad integradora 1: Lenguajes regulares y su procesamiento

Instrucciones generales: Lea cuidadosamente la descripción paso a paso de la actividad integradora. Esta actividad se debe realizar mediante los equipos del reto definidos previamente. Una vez completada su actividad, suba a Canvas , por equipo, un archivo ZIP con el código fuente de su programa y un archivo PDF con el reporte global de la actividad.

Objetivo general: Diseñar una aplicación en Python que dada una expresión regular (R) genere el autómata finito determinista (AFD) correspondiente que reconoce el lenguaje asociado, mostrando gráficamente el AFD en una aplicación para su correspondiente uso y verificación de funcionamiento.

Instrucciones particulares: A continuación se detallan las instrucciones específicas de esta actividad, comenzando con las entradas y salidas de la aplicación. Posteriormente, se detallan los pasos necesarios para convertir las entradas en las salidas.

- **Entradas:**
  1. **Alfabeto** (Σ): símbolos del alfaberto separados por comas en texto plano. Por ejemplo, los alfabetos $\Sigma_1 = \{0, 1\}$, $\Sigma_2 = \{a, b, c\}$, y $\Sigma_3 = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f\}$ debe ser dados como: "0,1", "a,b,c", y "0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f". La cadena deberá ser proporcionada a su programa por el usuario a través de un caja de texto en una interfaz gráfica de usuario (por ejemplo, una página HTML o por línea de comandos en la terminal).
  2. **Expresión regular** (R): definir una expresión regular para el lenguaje que desea reconocer. Solo podrá hacer uso de las operaciones: (i) estrella de Kleene ($*$); (iii) unión ($|$); (iii) concatenación (.); (iv) paréntesis para agrupación de subexpresiones. Así entonces, algunos ejemeplos de expresiones regulares sobre $\Sigma = \{0, 1\}$ son: (a) $R_1 = 0.1.0.1*$, $R_2 = ((0.1*)|(1.1.1))*$.
- **Salidas:** El AFD correspondiente a la expresión regular, mostrado en una interfaz gráfica de usuario (UI, por sus siglas en inglés). Este AFD podrá luego ser alimentado con palabras para determinar si son reconocidas o no, es decir, si son parte o no del lenguaje representado por la expresión regular.

1. **Construcción del autómata finito no determinista.** Tome las entradas del usuario Σ y R. Luego, transforme R en un autómata finito no determinista (AFN) mediante las técnicas de construcción vistas en clase. También puede revisar el algoritmo de McNaughton-Yamada-Thompson para una mayor referencia. Puede representar la expresión mediante un árbol sintáctico para su procesamiento.
2. **Construcción del autómata finito determinista.** Tome el AFN de la fase anterior como entrada en esta fase. Deberá transformar el AFN en un AFD mediante el algoritmo de conversión visto en clase (algoritmo de subconjuntos).
3. **Presentación visual.** Tome el AFD de la fase anterior y represéntelo gráficamente, es decir, dibuje en una UI el diagrama de transición (use módulos de Python).
4. **Pruebas.** La UI de la fase anterior deberá contener una caja de texto donde se pueda introducir una palabra ($w = w_1 w_2 \cdots w_n$, donde cada $w_i \in \Sigma$) para que el AFD la procese. Además, agregue un botón para lanzar el procesamiento de la palabra. Ilustre visualmente cómo el AFD cambia de estado al procesar individualmente cada símbolo $w_i \in w$. Al terminar de procesar la palabra, muestre un mensaje anunciando si w es reconocida o no por el AFD.

Restricciones: Los algoritmos de conversión de expresión regular a AFN y de AFN a AFD deberán ser programados por usted en su totalidad. Pueden usar libremente módulos de Python que apoyen el desarrollo pero no dependa totalmente de ellos. Use sus conocimientos para construir la UI. Si no tiene experiencia haciendo UIs, use la terminal como medio para alimentar el programa, pero sí debe mostrar el AFD en una figura.

Evaluación: La evaluación será de manera presencial con el profesor disruptivo mediante una muestra del funcionamiento del programa.